Proyecto de programación en ensamblador Estructura de Computadores

- El proyecto consiste en la programación, en ensamblador del Motorola 88110, de un conjunto de rutinas que realicen el ***filtrado de una imagen*** mediante un filtro programable.

- La imagen será una matriz de píxeles, cada uno de los cuales se representa mediante un ***byte sin signo*** que especifica su nivel de gris (0 equivale a negro y 255 a blanco).

- El filtro está basado en una operación recursiva de convolución con un núcleo representado por una matriz de 3x6 valores enteros que define una matriz 3x3 de coeficientes fraccionarios:
  - Cada elemento del filtro se representa por un par de números enteros (N, D) cuyo ***cociente*** determina el correspondiente elemento de la matriz 3x3 de coeficientes fraccionarios.

Enunciado feb-jul 2020/2021

- El enunciado de este proyecto **está basado en el planteado el curso pasado**, por lo que aquellos alumnos que tengan que repetir o corregir el proyecto que desarrollaron entonces, podrán partir de los programas ya realizados anteriormente. Sin embargo, algunas o todas las pruebas del proyecto que se realizarán en este curso serán diferentes, por lo que, en caso necesario, *deberán adaptar la implementación de las subrutinas de modo que superen las pruebas que se establezcan, tanto para la convocatoria de febrero, como para la de julio*.

- Por otra parte debe observar con atención el apartado de este documento que describe las normas de entrega y, en particular, la especificación del contenido que debe incluir la memoria del proyecto y las fechas de entrega de los ***hitos evaluables***.

- El hecho de plantear un proyecto similar al del curso pasado tiene además las siguientes implicaciones:
  - Los alumnos que ya hubieran formado parte de un grupo durante convocatorias anteriores y hubieran realizado al menos una entrega **solo podrán establecer grupo con el mismo compañero** de dicha convocatoria o, alternativamente, realizar el proyecto de **forma individual**.

  - Se realizará una revisión minuciosa de los proyectos realizados en este semestre para descartar o localizar posibles **casos de copia** que desafortunadamente se siguen produciendo (y detectando) en la mayoría de las convocatorias.

Definición del filtro

El filtro es una matriz cuadrada de orden 3 cuyos elementos son fraccionarios y están definidos por una matriz de 18 coeficientes enteros con signo.

Ejemplo:

Filtro: 1, 8, 1, 8, 1, 8,
1, 8, 0, 8, 1, 8,
1, 8, 1, 8, 1, 8

Matriz de filtro:

1/8	1/8	1/8	0,125	0,125	0,125
1/8	0/8	1/8	0,125	0,0	0,125
1/8	1/8	1/8	0,125	0,125	0,125

Cálculo del filtro de una imagen (1)

Im_original = (m[i,j]), Filtro = (f[i,j]), Im_filtrada = (r[i,j])

- r[i,j] = f[0,0]·m[i-1,j-1] + f[0,1]·m[i-1,j] + f[0,2]·m[i-1,j+1] +
  + f[1,0]·m[i,j-1] + f[1,1]·m[i,j] + f[1,2]·m[i,j+1] +
  + f[2,0]·m[i+1,j-1] + f[2,1]·m[i+1,j] + f[2,2]·m[i+1,j+1]

Im_original

1	2	3	4	5	6	7	...
5	6	7	8	9	0	1	...
9	0	1	2	3	4	5	...
3	4	5	6	7	8	9	...
7	8	9	0	1	2	3	...
.....							

Filtro

0,125	0,125	0,125
0,125	0,0	0,125
0,125	0,125	0,125

Valor de m[2,2]=1 Valor de r[2,2]=4,75 → 4

Cálculo del filtro de una imagen (2)

Filtro		
0,125	0,125	0,125
0,125	0,0	0,125
0,125	0,125	0,125

MFiltro		
1 8	1 8	1 8
1 8	0 8	1 8
1 8	1 8	1 8

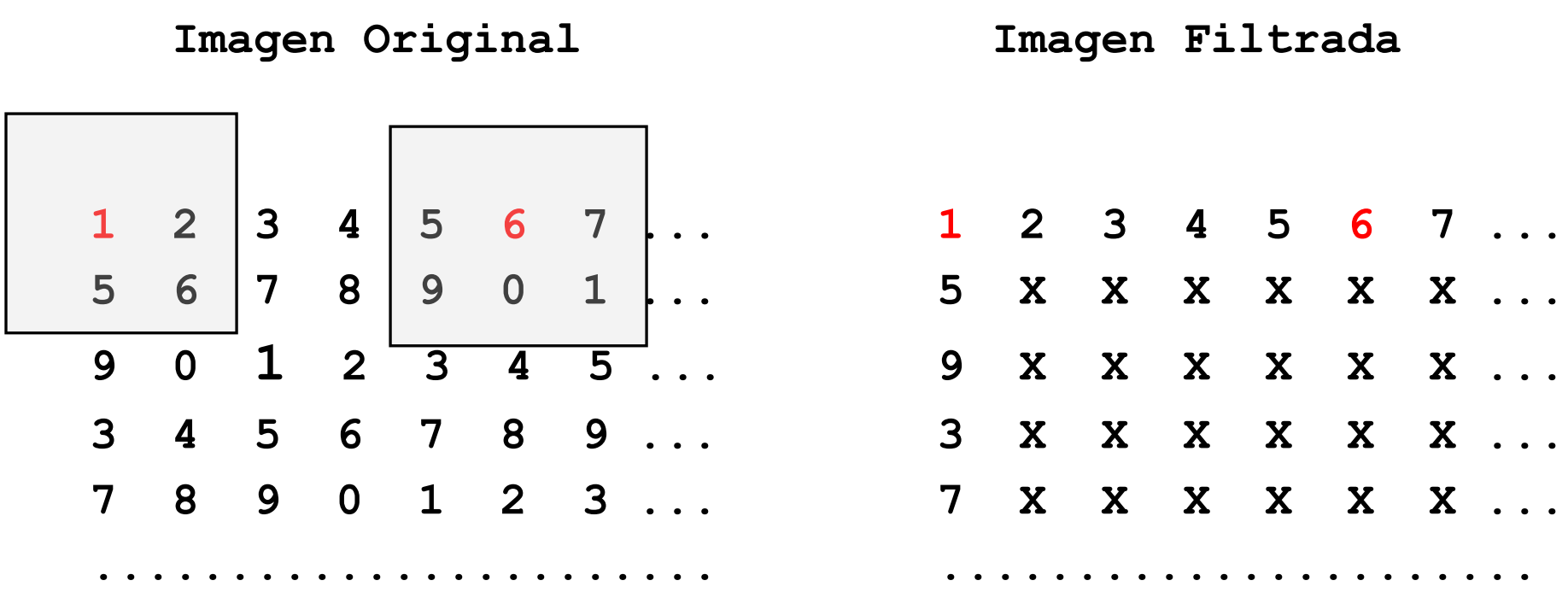
Subimagen		
6	7	8
0	1	2
4	5	6

- Se opera con la matriz **MFiltro** definida por sus 18 elementos enteros
- Se utiliza aritmética entera con signo: "**add**", "**muls**" y "**divs**"

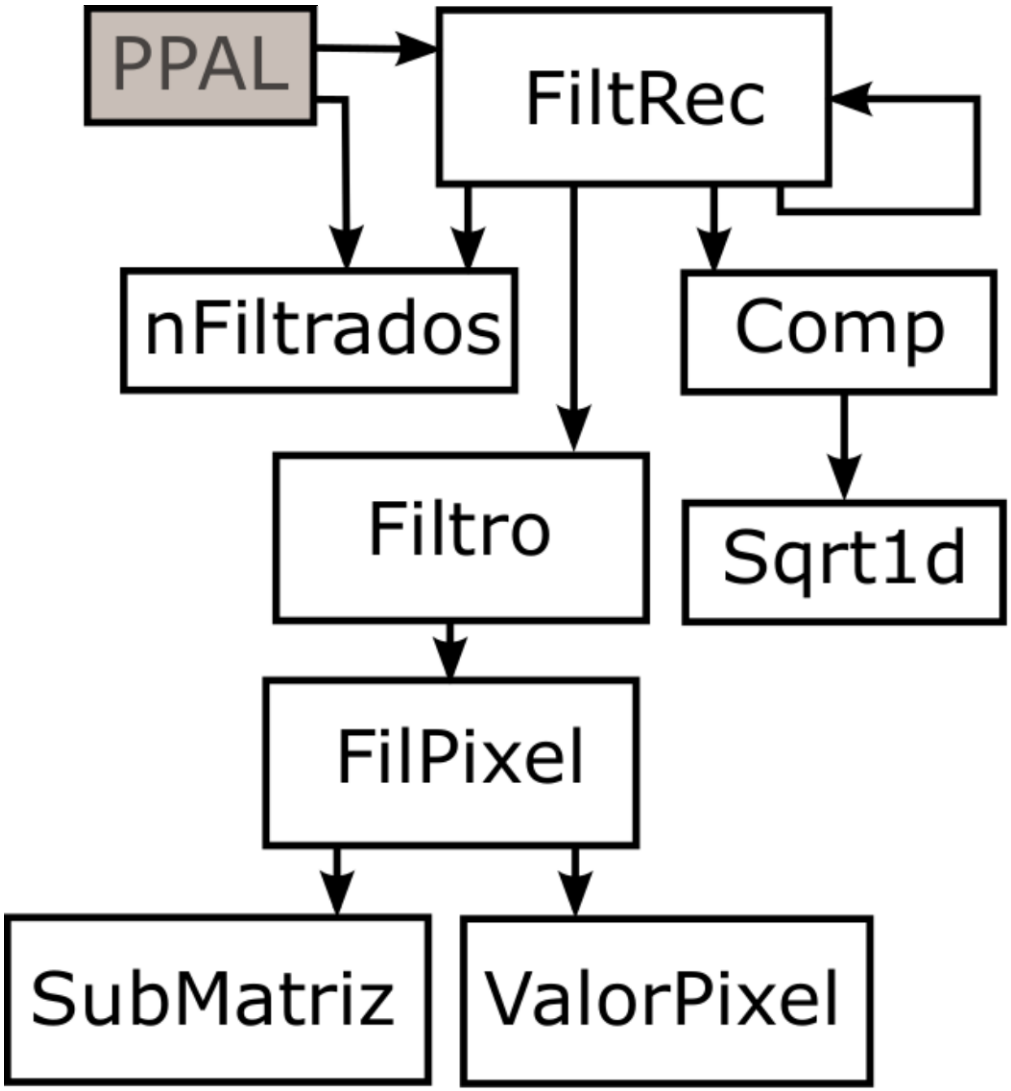
```
r[2,2] = (6/8 +7/8 +8/8 +0 +0 +2/8 +4/8 +5/8 +6/8) = 1

r[2,2]=256*(6/8 +7/8 +8/8 +0 +0 +2/8 +4/8 +5/8 +6/8)/256 =
      =(256*6/8 + 256*7/8 + ... + 256*5/8 + 256*6/8)/256 = 4
```

Filtrado de los bordes de una imagen



Jerarquía de las rutinas



Número de filtrados

```
NFiltrados = nFiltrados (oper)
```

- Parámetros (en la pila):

  oper: De entrada. Se pasa por valor. Indica qué operación debe realizar la subrutina.

- Valor de retorno:

  NFiltrados: Devuelve en r29 el valor de la variable estática nF.

- Algoritmo:

  1. Si oper >= 0 → Inicializa la variable estática nF al valor oper.
  2. Si oper < 0 → Incrementa la variable estática nF.
  3. Retorna cargando NFiltrados = nF en r29.

Raíz cuadrada entera

```
rc = Sqrt1d (Num)
```

- Parámetros (en la pila):

  Num: De entrada. Se pasa por valor.

- Valor de retorno:

  rc: Devuelve en r29 una aproximación entera de la raíz cuadrada de Num x 100.

- Algoritmo:

  1. Si Num < 2 → retorna cargando rc = Num x 10 en r29.
  2. a = Num x 100; b = 1.
  3. a) a = (a + b)/2; b = (Num x 100)/a.

     b) si b > a → a ↔ b.

     c) Si (a – b) > 1 → vuelve al paso 3 a).
  4. Retorna cargando rc = b en r29;

Compara imágenes

```
Diferencia = Comp (Imagen1, Imagen2)
```

- Parámetros (en la pila):

  Imagen1: De entrada. Se pasa por dirección: M (1 palabra), N (1 palabra), MxN elementos (1 byte cada uno).

  Imagen2: De entrada. Se pasa por dirección: M (1 palabra), N (1 palabra), MxN elementos (1 byte cada uno).

- Valor de retorno:

  Diferencia: Devuelve en r29 el valor de la diferencia entre las imágenes.

- Algoritmo:

  1. Inicializa a cero un acumulador de diferencias (*Dif*).
  2. Recorre los MxN píxeles de cada matriz incrementando Dif. en el cuadrado de la diferencia: Dif. = Dif. + $(Imagen1_{ij} – Imagen2_{ij})^2$.
  3. Asigna Diferencia = Sqrt1d(Dif).
  4. Retorna cargando Diferencia en r29.

Valor del píxel filtrado (1)

```
VPixel = ValorPixel (SubImg, MFiltro)
```

- Parámetros (en la pila):

  SubImg: De entrada. Se pasa por dirección: 9 elementos (1 byte sin signo cada uno).
  Es la submatriz de 3x3 centrada en el píxel a filtrar.

  MFiltro: De entrada. Se pasa por dirección. Formada por:

  – 18 valores correspondientes a los nueve pares de elementos (enteros con signo) de la matriz 3x6 que definen el núcleo de filtrado.

- Valor de retorno:

  VPixel: Devuelve en r29 el valor provisional del píxel filtrado (entero con signo de 32 bits) .

- Descripción:

  – Aplica el filtro MFiltro a la submatriz SubImg.

Valor del píxel filtrado (2)

```
VPixel = ValorPixel (SubImg, MFiltro)
```

- Algoritmo
  1. Inicializa a cero el acumulador ACC.
  2. Inicializa un puntero al primer elemento de SubImg y otro al primer elemento de la matriz de filtro.
  3. Recorre, en un bucle, los 9 pares de elementos de la matriz de filtro y los 9 elementos de SubImg multiplicándolos entre sí y acumulando el resultado. En cada iteración:
     - Pasa el valor del píxel sin signo a un registro **`rxx`**
     - Multiplica **`rxx`** por 256 y por el coeficiente N del filtro, divide entre el coeficiente D del filtro y acumula en ACC
  4. Divide ACC por 256.
  5. Asigna r29 = ACC y retorna al llamante.

Extracción de Submatriz (1)

`SubMatriz (Imagen, SubImg, i, j)`

- Parámetros (en la pila):

  Imagen: De entrada. Se pasa por dirección: M (1 palabra), N (1 palabra), MxN elementos (1 byte cada uno). Es la imagen de la que se debe extraer una submatriz de 3x3 elementos.

  SubImg: De salida. Se pasa por dirección: 9 elementos (1 byte sin signo cada uno). Es la matriz de 3x3 elementos en la que se debe copiar la submatriz de `Imagen` centrada en el pixel [`i`, `j`].

  i: Número de fila. (1..M-2). De entrada. Se pasa por valor.

  j: Número de columna. (1..N-2). De entrada. Se pasa por valor.

- Descripción:
  - Copia la submatriz de Imagen centrada en el píxel [i, j] sobre los 9 píxeles de SubImg.

Filtro de un píxel (1)

```
VPixel = FilPixel (Imagen, i, j, MFiltro)
```

- Parámetros (en la pila):

  Imagen: De entrada. Se pasa por dirección: M (1 palabra), N (1 palabra), MxN elementos (1 byte cada uno). Es la imagen que contiene el píxel a filtrar.

  i: Número de fila. De entrada. Se pasa por valor.

  j: Número de columna. De entrada. Se pasa por valor.

  MFiltro: De entrada. Se pasa por dirección. Formada por:

  – 18 valores correspondientes a los nueve pares de elementos (enteros con signo) de la matriz 3x6 que definen el núcleo de filtrado.

- Valor de retorno:

  VPixel: Devuelve en r29 el valor del píxel filtrado (entero sin signo en el rango (0..255).

- Descripción:

  – Aplica la máscara de filtrado al píxel [i,j] de la imagen.

Filtro de un píxel (2)

```
VPixel = FilPixel (Imagen, i, j, MFiltro)
```

- Algoritmo
  1. Determina si el píxel [i, j] pertenece al borde de la Imagen, es decir, si i=0 o j=0 o i=M-1 o j=N-1.
  2. Si pertenece al borde: Copia en r29 el valor del píxel [i, j] y continúa en el paso 7.
  3. Reserva espacio en el marco de pila para almacenar una submatriz de 3x3 bytes sin signo (SubImg) (3 palabras).
  4. Llama a SubMatriz (Imagen, SubImg, i, j).
  5. Llama a r29=ValorPixel (SubImg, Mfiltro).
  6. Si r29 < 0 se ajusta a 0. Si r29 > 255 se ajusta a 255.
  7. Retorna al llamante teniendo r29 el valor del píxel [i, j] filtrado.

Filtro de una imagen (1)

`Filtro (Imagen, ImFiltrada, MFiltro)`

- Parámetros (en la pila):

  Imagen: De entrada. Se pasa por dirección: M (1 palabra), N (1 palabra), MxN elementos (1 byte cada uno). Es la imagen que se debe filtrar.

  ImFiltrada: De salida. Es la imagen que resulta de aplicar el filtro a la imagen de entrada. Se pasa por dirección: M (1 palabra), N (1 palabra), MxN elementos (1 byte cada uno).

  MFiltro: De entrada. Se pasa por dirección. Formada por:

  – 18 valores correspondientes a los nueve pares de elementos (enteros con signo) de la matriz 3x6 que definen el núcleo de filtrado.

- Descripción:

  – Aplica la máscara de filtrado definida por MFiltro a la imagen de entrada. Deja el resultado en la dirección definida por ImFiltrada.

Filtro de una imagen (2)

```
Filtro (Imagen, ImFiltrada, MFiltro)
```

- Algoritmo:
  1. Copia M y N sobre la imagen filtrada.
  2. Desde i=0 hasta i=M-1
     - Desde j=0 hasta j=N-1
       1. Prepara parámetros y llama a FilPixel (Imagen, i, j, MFiltro).
       2. Almacena el valor de retorno en la posición [i,j] de la imagen filtrada.
  3. Retorna al llamante.

Filtro recursivo (1)

```
Ret = FiltRec(ImagenIn, ImagenOut, MFiltro, NCambios, MaxFiltrados)
```

- Parámetros (en la pila):

ImagenIn: De entrada. Se pasa por dirección: M (1 palabra), N (1 palabra), MxN elementos (1 byte cada uno). Es la imagen de entrada a la que se ha de aplicar el filtro.

ImagenOut: De salida. Se pasa por dirección: M (1 palabra), N (1 palabra), MxN elementos (1 byte cada uno). Es la imagen que resulta de aplicar el filtro.

MFiltro: De entrada. Se pasa por dirección. Formada por:

- 18 valores correspondientes a los nueve pares de elementos (enteros con signo) de la matriz 3x6 que definen el núcleo de filtrado.

NCambios: De entrada/salida y se pasa por valor (1 palabra). Es el valor de la diferencia entre la imagen de entrada y la filtrada que determina si se debe hacer o no una nueva llamada recursiva. Al finalizar FiltRec, NCambios deberá contener la diferencia obtenida tras la última operación de filtrado.

Filtro recursivo (2)

```
Ret = FiltRec(ImagenIn, ImagenOut, MFiltro, NCambios, MaxFiltrados)
```

- Parámetros (en la pila):

  MaxFiltrados: De entrada/salida y se pasa por valor (1 palabra). Es el número máximo de filtrados que se debe realizar. Al finalizar FiltRec, MaxFiltrados deberá contener el número de operaciones de filtrado realizadas.

- Valor de retorno:

  Ret: Devuelve 0 en r29 si la operación de filtrado se realizó correctamente, o bien -1 indicando que se alcanzó el número máximo de filtrados.

- Descripción:
  - Realiza filtrados de una imagen de entrada ImagenIn de forma recursiva, dejando el resultado final en ImagenOut.

- Condiciones de salida de la recursividad:
  1. La diferencia entre la imagen original y la filtrada es **menor** que NCambios.
  2. NFiltrados = MaxFiltrados (NFiltrados es el valor devuelto por la subrutina nFiltrados a la que se llama con el parámetro oper < 0).

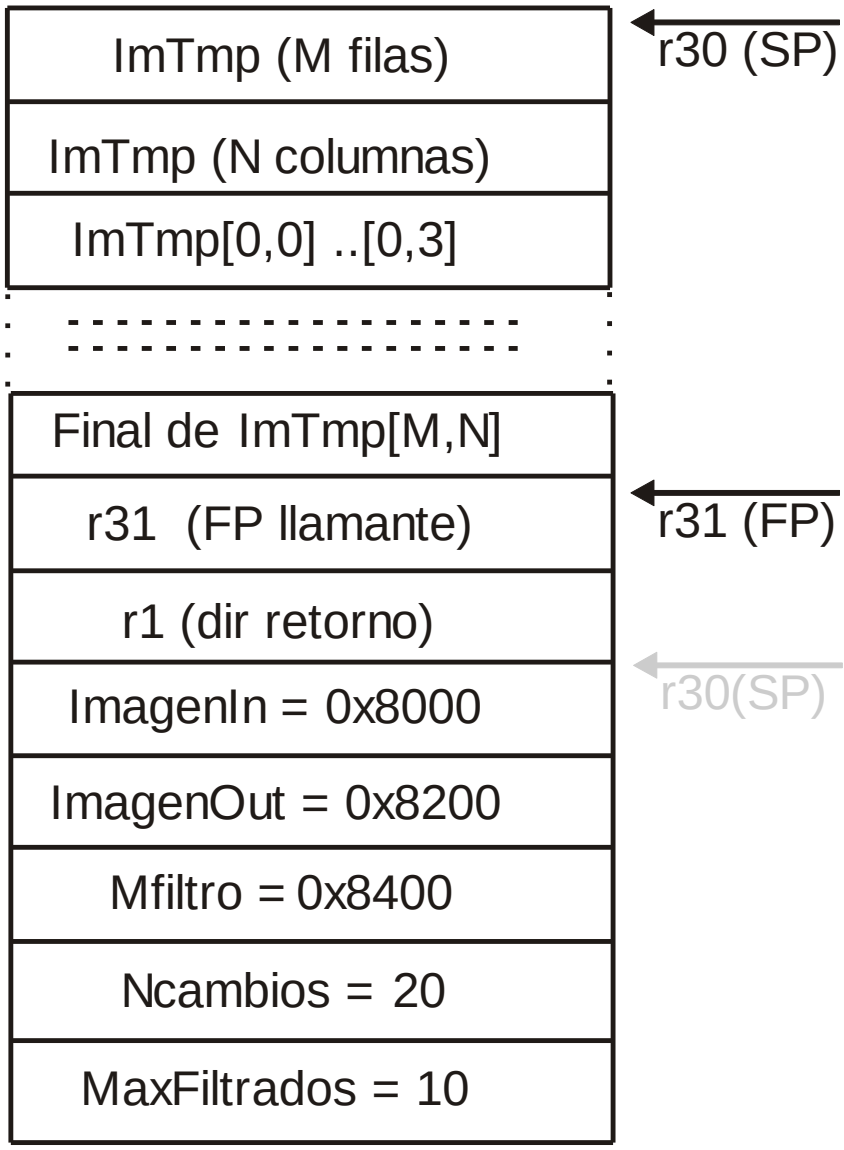
Filtro recursivo (3)

```
Ret = FiltRec(ImagenIn, ImagenOut, MFiltro, NCambios, MaxFiltrados)
```

- Algoritmo:
  1. Reserva espacio en el marco de pila para la variable temporal ImagenTmp: 4+4+M·N ajustado por exceso a múltiplo de 4.
  2. Llama a Filtro (ImagenIn, ImagenOut, Mfiltro).
  3. Copia la imagen filtrada (ImagenOut) en la variable temporal ImagenTmp.
  4. Llama a Comp pasando las imágenes original (ImagenIn) y filtrada (ImagenOut).
  5. Llama a la subrutina nFiltrados con el parámetro oper < 0.
  6. Actualiza en la pila MaxFiltrados y NCambios con los valores devueltos por nFiltrados y Comp.
  7. Si r29 < (NCambios de entrada) asigna r29 = 0 y continúa en el paso 9.
     Si NFiltrados = (MaxFiltrados de entrada) asigna r29 = -1 y continúa en el paso 9.
  8. Llama a FiltRec pasando la copia de la imagen filtrada (ImagenTmp), la imagen de salida (ImagenOut), Mfiltro, NCambios y MaxFiltrados. (Al retornar de esta llamada recursiva el valor devuelto en r29 es 0 o -1 dependiendo del número de filtrados realizados. La imagen filtrada, ImagenOut, se habrá actualizado con el resultado de la última operación de filtrado).
     Propaga los parámetros de salida NCambios y MaxFiltrados.
  9. Retorna al llamante.

Tratamiento de la pila

```
FiltRec:
      PUSH    (r1)
      PUSH    (r31)
      addu    r31,r30,r0
      ld      r10,r31,8
      .......
```



Ejemplo: Filtro de un píxel

Imagen:

44	44	44	44	44	44	44	44
44	34	34	33	44	44	44	44
44	44	88	44	44	44	44	44
44	44	44	44	44	44	44	44

Mfiltro: (k=256)

1 8	1 8	1 8
1 8	0 8	1 8
1 8	1 8	1 8

k*34/8 + k*34/8 + k*33/8 +
k*44/8 + k*00 + k*44/8 +
k*44/8 + k*44/8 + k*44/8 = 3DE0

3DE0/k = 3D

Ejemplo: Filtro de un píxel

```
r30=36848 (0x8FF0)
Direcciones de memoria:
36848   00800000  02000000  02000000  30800000

32768   04000000  08000000  44444444  44444444
32784   44343433  44444444  44448844  44444444
32800   44444444  44444444

32816   01000000  08000000  01000000  08000000
32832   01000000  08000000  01000000  08000000
32848   00000000  08000000  01000000  08000000
32854   01000000  08000000  01000000  08000000
32870   01000000  08000000

Resultado:
r30=36848 (0x8FF0)  r29=61 (0x3D)
```

Ejemplo: Filtro recursivo

Entrada:

Imagen:

```
FF  01  02  FF  FF  03  04  FF
FF  05  06  FF  FF  07  08  FF
FF  05  06  FF  FF  07  08  FF
FF  01  02  FF  FF  03  04  FF
```

Filtro:

```
1 8  1 8  1 8
1 8  0 8  1 8
1 8  1 8  1 8
```

MaxFiltrados = 3 **Ncambios = 0**

Ejemplo: Filtro recursivo

```
                org   0x8000
IMAGEN:         data  4, 8
                data  0xFF0201FF, 0xFF0403FF
                data  0xFF0605FF, 0xFF0807FF
                data  0xFF0605FF, 0xFF0807FF
                data  0xFF0201FF, 0xFF0403FF

FILTRO:         data  1, 8, 1, 8, 1, 8
                data  1, 8, 0, 8, 1, 8
                data  1, 8, 1, 8, 1, 8

                org   0x8100
FILTRADA:       res   40
```

Ejemplo: Filtro recursivo

```
        org   0x8400
ppal:   or    r30, r0, 0x9000
           ...........
        LEA   (r10, IMAGEN)
        LEA   (r11, FILTRADA)
        LEA   (r12, FILTRO)
        or    r13, r0, r0       ; NCambios=0
        addu  r14, r0, 3        ; máx: 3 filtrados
        PUSH  (r14) ; MaxFiltrados
        PUSH  (r13) ; Ncambios
        PUSH  (r12) ; FILTRO
        PUSH  (r11) ; FILTRADA
        PUSH  (r10) ; IMAGEN
        bsr   FiltRec
            ...........
```

Ejemplo: Filtro recursivo

```
r30=36844 (0x8FEC)

Direcciones de memoria:

00000   00000000

32768   04000000  08000000  FF0102FF  FF0304FF
32784   FF0506FF  FF0708FF  FF0506FF  FF0708FF
32800   FF0102FF  FF0304FF

32800                       01000000  08000000
32816   01000000  08000000  01000000  08000000
32832   01000000  08000000  00000000  08000000
32816   01000000  08000000  01000000  08000000
32816   01000000  08000000  01000000  08000000

36832                                 00800000
36848   00810000  28800000  00000000  03000000
```

Ejemplo: Filtro recursivo

```
Resultado:
   FF  01  02  FF  FF  03  04  FF
   FF  8B  73  93  93  75  8C  FF
   FF  8B  73  93  93  75  8C  FF
   FF  01  02  FF  FF  03  04  FF


r29=-1 (0x0FFFFFFF)              nF=3

NCambios = 0xC1            MaxFiltrados = 3
```

Ejemplo: Filtro recursivo

```
 Resultado:
r30=36844 (0x8FEC) r29=-1 (0xFFFFFFFF)

Direcciones de memoria

00000   03000000

32768   04000000  08000000  FF0102FF  FF0304FF
32784   FF0506FF  FF0708FF  FF0506FF  FF0708FF
32800   FF0102FF  FF0304FF

33024   04000000  08000000  FF0102FF  FF0304FF
33040   FF8B7393  93758CFF  FF8B7393  93758CFF
33056   FF0102FF  FF0304FF

36832                                 00800000
36848   00810000  28800000  C1000000  03000000
```

Programas de prueba

```
LEA:  MACRO...

        org   xxx
D1:     data...
D2:     data...
D3:     res...

        org   yyy
PPAL: "inicialización de la pila"
        ...
        "paso de parámetros (PUSH)"
        ...
        bsr subrutina
        "vaciado de la pila (POP)".
        stop
```

Entrega (fechas)

CONVOCATORIA DE FEBRERO 2021

El plazo de entrega del proyecto estará abierto para entregar código y memoria:
Desde el martes día 20 de octubre hasta el lunes día 21 de diciembre de 2020.

Cada grupo podrá disponer de las siguientes correcciones:

- Primera corrección: **Viernes 23/10/2020 18:00**
- 1 corrección adicional: 26, 27, 28 ó 29 de octubre
- 1er hito evaluable : Viernes 30/10/2020:
  subrutinas nFiltrados y Sqrt1d: **1 punto**
- 1 corrección adicional: 3, 4, 5, 6, 10, 11 ó 12 de noviembre
- 2º hito evaluable : Viernes 13/11/2020:
  subrutinas del 1er hito más Comp, ValorPixel y SubMatriz: **1 punto**
- 4 correcciones adicionales: 16, 17, 18, 19, 20, 23, 24, 25, 26, 27 y 30 de noviembre,
  1, 2, 3, 4, 9, 10, 11, 14, 15, 16, 17 y 18 de diciembre
- Última corrección: Lunes 21/12/2020

Todas las correcciones se realizarán a partir de las 21:00, salvo la primera del viernes 23 de octubre, que se realizará a las 18:00. Para solicitar una corrección bastará con entregar correctamente los ficheros del proyecto antes de dicha hora límite.

El examen del proyecto está planificado para el jueves día 21 de enero de 2021.

Entrega (contenido)

Ficheros:

1. **`autores (solo en alta de grupo)`**: Es un fichero ASCII que deberá contener los apellidos, nombre, número de matrícula, DNI y dirección de correo electrónico de los autores del proyecto. El proyecto se realizará individualmente o en grupos de dos alumnos. Cada línea de este fichero contendrá los datos de uno de los autores de acuerdo al siguiente formato:

   `No Matrícula; DNI; apellido apellido, nombre; correo electrónico`

   NOTA: este fichero solo se entrega en el momento de hacer el registro en el Gestor de Prácticas. Después de ese momento se accederá mediante el par usuario:clave utilizados durante el registro

2. **`filtror21.ens`**: Contendrá las subrutinas que componen el proyecto debidamente comentadas junto con un programa principal y los datos de prueba para cada una de ellas que se haya utilizado para su depuración.

3. **`memoria.pdf:`** *Memoria*, en formato PDF y tamaño DINA4, en cuya portada deberá figurar claramente el nombre y apellidos de los autores del proyecto, identificador del grupo de alumnos (el mismo que emplean para realizar las entregas y consultas) y el nombre de la asignatura.

Entrega (memoria)

La memoria debe contener los siguientes puntos:

- Histórico del desarrollo de las rutinas, con fechas, avances y dificultades encontradas, especificando el trabajo que realiza cada miembro del grupo o si dicho trabajo es común. Se detallará en este apartado:
  - Número total de horas invertidas en el proyecto por cada miembro del grupo.
  - Relación de visitas realizadas a los profesores del proyecto.
- Descripción *resumida* del juego de ensayo (conjunto de casos de prueba) que el grupo haya diseñado y utilizado para probar el correcto funcionamiento del proyecto. Una única prueba por cada subrutina.
- Observaciones finales y comentarios personales de este proyecto, entre los que se debe incluir una descripción de las principales dificultades surgidas para su realización.

Evaluación (2020/2021)

- El proyecto consta de tres partes: código y memoria (***código-memoria)***, pruebas de funcionamiento (***pruebas)*** *y* examen del proyecto ***(examen)***. Para superar el proyecto se deberá obtener la calificación de **apto en cada una de las tres partes**.

- Para que un grupo supere la parte de pruebas será necesario obtener al menos 5 puntos en la última corrección (o con anterioridad). La puntuación de esta parte es la siguiente:

  Hitos evaluables, **2 puntos** condicionados al logro de los hitos en las fechas anteriormente indicadas :

  – ***A****: Superar todas las pruebas de las subrutinas* **`nFiltrados`** y **`Sqrt1d`** ***(1 punto)***.

  – ***B****: Superar todas las pruebas de las subrutinas* **`nFiltrados, Sqrt1d, Comp, ValorPixel`** *y* **`SubMatriz`** ***(1 punto)***.

  Subrutina **`FilPixel`, 1 punto.** Proporcional al número de pruebas superadas.

  Subrutina **`Filtro`, 1 punto.** Proporcional al número de pruebas superadas.

  Subrutina **`FiltRec`, 6 puntos.** Proporcional al número de pruebas superadas.

- Examen: A partir de **3 puntos** hará media con la nota de pruebas.

- Código-memoria: **±1 punto** en función de la calidad. "**No apto**" requisitos básicos "Avisos importantes" pág. 20.

Evaluación (2020/2021)

- Calificación final:
  - pruebas >= 5 puntos; examen >= 3 puntos; código-memoria Apto:
    0,7 * pruebas + 0,3 * examen ± 1 (código-memoria)
  - examen < 3 puntos, código-memoria Apto: **Nota del examen**
  - pruebas < 5 puntos o código-memoria "No apto": **< 2 puntos**
- Nota del proyecto compensable con la nota de teoría: >= **2 puntos**

Ejemplo con imagen real

- Procesada con una versión escrita en C que realiza las mismas operaciones que el código del proyecto.
- La implementación en C tiene la misma estructura que la descrita en este proyecto
- Los valores de la matriz de filtro utilizada son:

MFiltro:

$$\begin{bmatrix} 3\ 15 & 3\ 15 & 3\ 15 \\ 0\ 15 & 15\ 15 & 0\ 15 \\ -3\ 15 & -3\ 15 & -3\ 15 \end{bmatrix}$$

definida como:

```
MFiltro: {3, 15, 3, 15, 3, 15, 0, 15, 15, 15, 0, 15, -3, 15, -3, 15, -3, 15}
```

Imagen original:

Imagen procesada (MFiltro: {3,15,3,15,3,15, 0,15,15,15,0,15, -3,15,-3,15,-3,15} 10 filtrados):